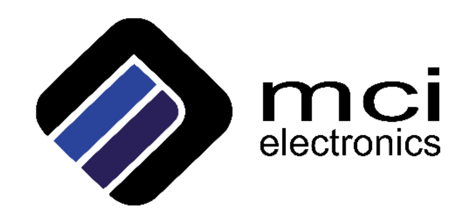
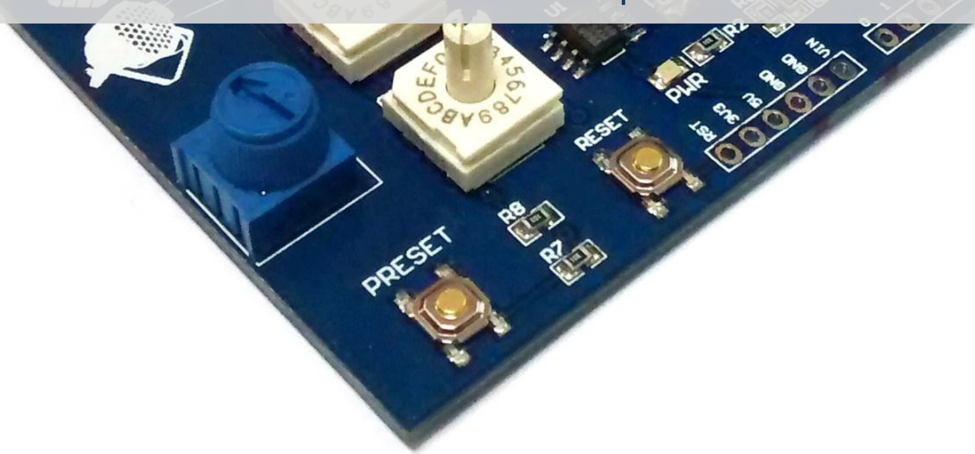





# MANUAL DE USUARIO

## ARDUINO DMX MASTER SHIELD

**MCI-TDD-01588 | REV. 1.0**




Ingeniería MCI Ltda.

Luis Thayer Ojeda 0115 of. 1105, Providencia, Santiago, Chile.

+56 2 23339579 | www.olimex.cl | info@olimex.cl

**Ingeniería MCI Ltda.**

Luis Thayer Ojeda 0115 Oficina 1105
Providencia, Santiago, Chile

www.olimex.cl
info@olimex.cl

Tel: +56 2 23339579
Fax: +56 2 23350589



***Atención:*** *cambios y modificaciones hechas en el dispositivo, no autorizados expresamente por MCI, anularán su garantía.*

Código Manual: MCI–MA-1026

## CONTENIDO

## INTRODUCCIÓN

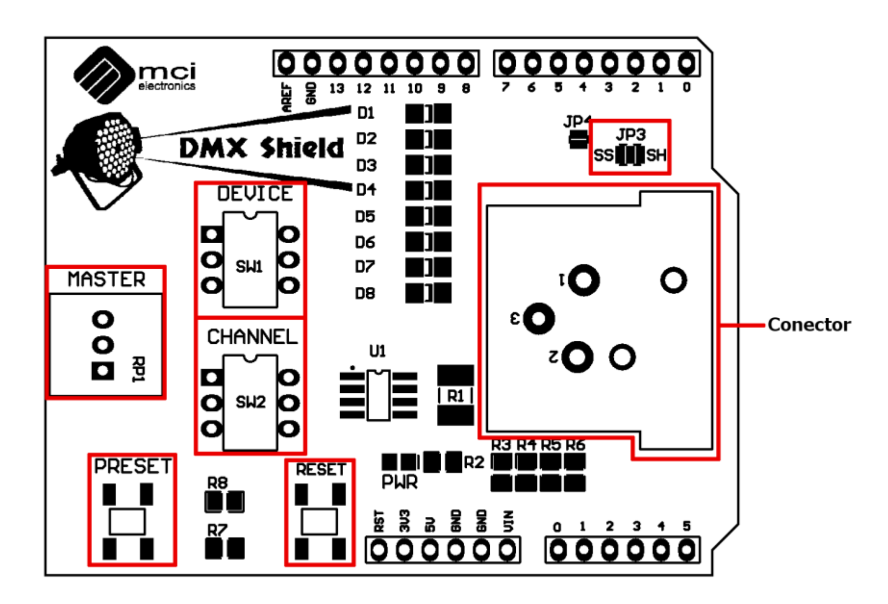
Esta tarjeta nos permite extender la funcionalidad de Arduino controlando focos de luces con protocolo DMX, generando las secuencias que deseemos. Ideal para eventos y fiestas.

El DMX Shield consta de un conector Amphenol y un chip transceptor para el protocolo RS-485, para la conexión de dispositivos basados en dicho protocolo de comunicación. También posee los conectores header comunes para su montaje sobre una placa Arduino compatible, un par de selectores codificados, un potenciómetro, un pulsador.

La placa DMX Shield necesita sólo estar montado sobre una placa Arduino, siendo muy sencilla la programación de efectos de luces y control mediante esta plataforma.

## PARTES DEL DISPOSITIVO

Descripción de las partes más importantes de la tarjeta Arduino DMX Master Shield.




- **MASTER:** Potenciómetro lineal conectado a la entrada análoga A0
- **DEVICE:** Selector mecánico de 16 posiciones
- **CHANNEL:** Selector mecánico de 16 posiciones
- **RESET:** Botón conectado al Reset de Arduino
- **PRESET:** Botón conectado a la entrada A5
- **JP3:** Selector entre el pin D1 y D3 para la comunicación
- **CONECTOR:** Para los focos con protocolo DMX

## DESCRIPCIÓN DEL PROTOCOLO DMX-512

El protocolo DMX, cuyo nombre es un acrónimo de Digital MultipleX, fue diseñado y se |

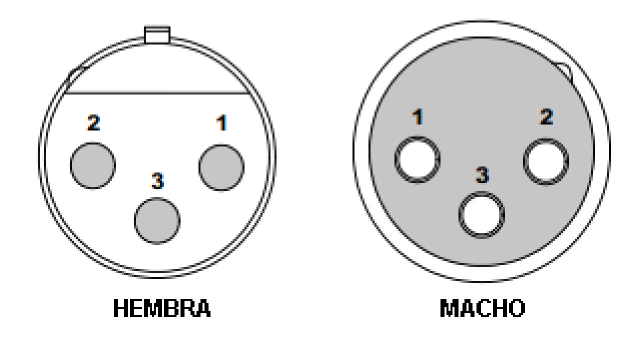



*Figura Nº 2: tipo de conector Amphenol (macho y hembra)*

En la figura Nº 2 se puede apreciar el tipo de conector utilizado para la conexión de dispositivos a través del protocolo. Este conector, utilizado en el DMX Shield, posee 3 pines, donde dos pines son de tipo diferencial y el tercero es la conexión a GND o tierra de alimentación.

Los canales de datos corresponden a una serie de registros que poseen los dispositivos del protocolo DMX-512. Por ejemplo, si un determinado foco de luz RGB (que puede proyectar diferentes brillos de los 3 colores primarios, rojo, verde y azul) debe ser programado mediante este protocolo, seguramente contará con 3 canales diferentes:

- El canal 1, para manejo del tono de color rojo.
- El canal 2, para manejo del tono de color verde.
- El canal 3, para manejo del tono de color azul.

Cada canal, según el protocolo, sólo puede tener valores entre 0 y 255 (es decir, 8 bits). En otro caso, si un foco de luz sólo proyecta luz monocromática, contará por lo menos con 1 canal, que determinará el brillo para el color base del foco. Otros canales pueden ser definidos por los fabricantes de los dispositivos para generar efectos de luz, operaciones especiales, etc.

Cuando dentro de una red DMX-512 (un universo de dispositivos, como se lo denomina en la formalización del protocolo) se conectan 1 o más dispositivos en serie, se hacen necesarias dos configuraciones adicionales:

- Acoplar un terminador resistivo de al final de último dispositivo conectado, para evitar las reflexiones de señal al final de la línea.
- Asignar un rango de canales diferentes a cada dispositivo diferente de la red.

Por ejemplo, supóngase lo siguiente:

Ejemplo 1: Se necesita acoplar 11 focos RGB de diferentes marcas y modelos a una red basada en DMX-512, donde los dispositivos tienen cada uno 3 canales de datos y estos deben configurarse en forma consecutiva. Esto se podrá realizar de la forma indicada en la figura Nº 3, que se explicará a continuación:

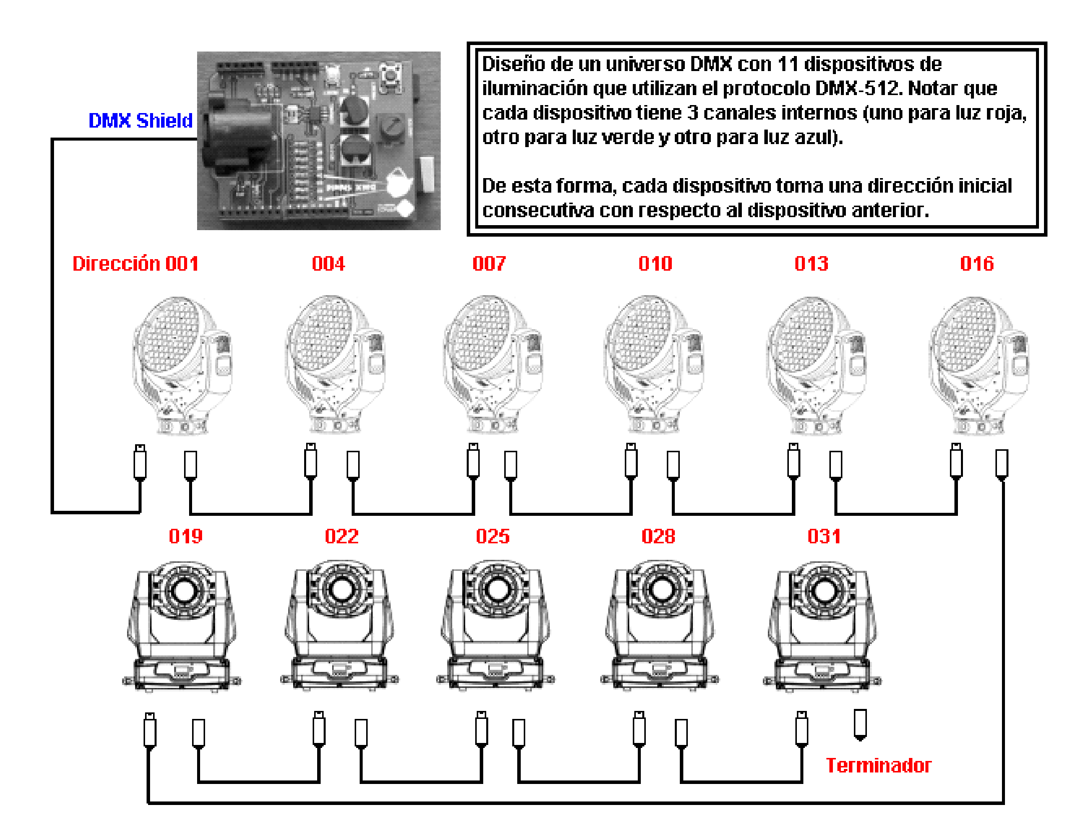



*Figura Nº 3: ejemplo de conexión de 11 focos para iluminación, regidos bajo el protocolo DMX-512. Cada foco tiene 3 canales de control, por lo que la dirección de cada foco es consecutiva respecto del anterior. Cada foco tendrá asignados 3 canales consecutivos.*

En el ejemplo recién visto, como cada foco dispone de 3 canales internos, cada uno de dichos focos tendrá un canal inicial separado de los focos contiguos por 3 unidades (focos con canales 001, 004, 007, 010, etc.). Internamente, cada foco tendrá asignados sus tres canales, mapeados de la forma siguiente:

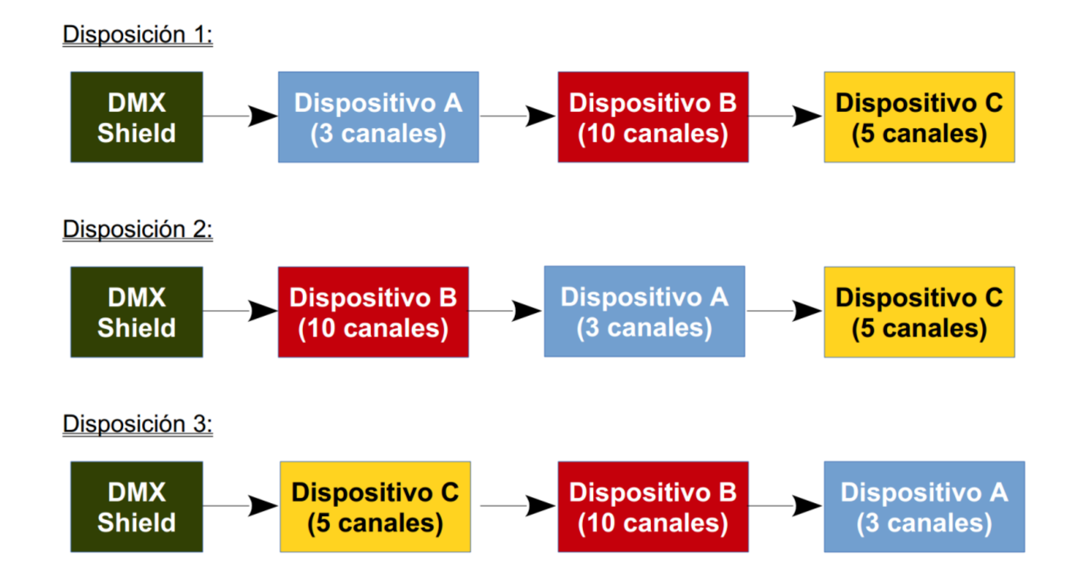
a) foco 001: canales 1, 2 y 3
b) foco 004: canales 4, 5 y 6
c) foco 007: canales 7, 8 y 9
d) foco 010: canales 10, 11 y 12.

Y así sucesivamente para cada foco siguiente. Como se ve, los canales se asignan de forma contigua para maximizar la presencia de dispositivos DMX-512 en la red de comunicaciones.

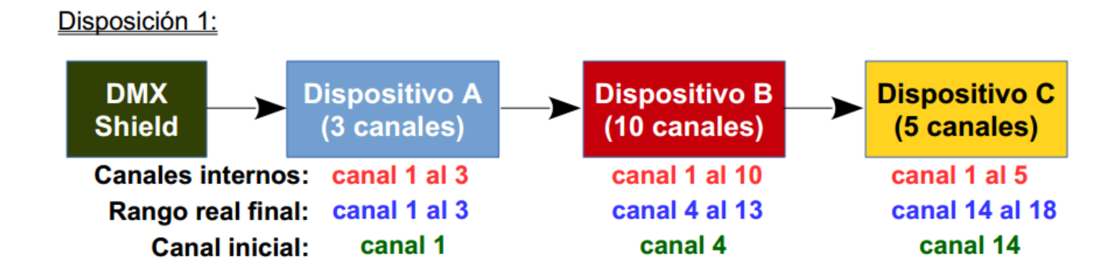
En este punto, es muy importante entender cómo funciona el protocolo DMX-512 y la asignación de canales para cada dispositivo. Para comprender correctamente este punto, se expondrá a continuación un ejemplo adicional, más completo:

Ejemplo 2: supóngase que ahora se tienen 3 dispositivos DMX, cada uno con diferente cantidad de canales: el dispositivo A, con 3 canales; el dispositivo B, con 10 canales; y, el dispositivo C, con 5 canales. Distribúyanse los tres dispositivos de diversas formas y calcúlense el rango de canales para cada dispositivo.

Para responder este problema, se realizarán las siguientes tres disposiciones de dispositivos:

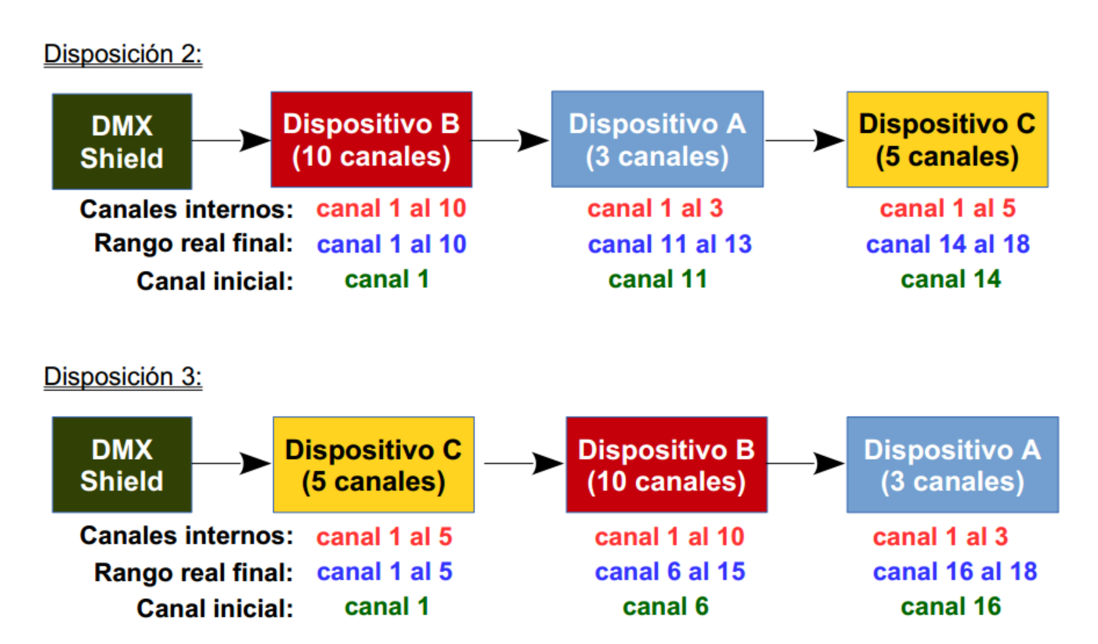



Para calcular los canales asignados a cada dispositivo en cada disposición se debe recordar que: a) el primer canal posible a asignar es el 1; b) el último canal posible a asignar es el 512; c) cada dispositivo tiene un canal inicial, a partir del cual se establecen los siguientes que necesita dicho dispositivo; d) dos dispositivos no deben compartir un mismo canal. De esta forma, y suponiendo una disposición contigua de canales desde el canal 1 (desde el DMX Shield hacia la derecha), se obtienen los siguientes cálculos:




Es importante fijarse en que cada dispositivo tiene una serie de canales internos (el rango interno de canales), pero que al conectar cada dispositivo dentro de la red DMX-512, esos rangos se traducen a un nuevo rango continuo. Fijarse también que, por orden, los canales del segundo dispositivo se mapean a continuación de los canales del primero, los del tercero a continuación del segundo, y así sucesivamente. También podría considerarse dejar rangos de canales no mapeados a ningún dispositivo como forma de insertar nuevos dispositivos a futuro.

Siguiendo con el ejemplo, para la disposición 2 y 3 se tendrán los siguientes resultados:




## PROGRAMACIÓN DE ARDUINO USANDO LA LIBRERÍA DMXSIMPLE

Para programar el DMX Shield se debe utilizar en el programa una serie de funciones que implementen el protocolo DMX-512. Por esta razón, para los ejemplos realizados, se utilizará una biblioteca de distribución libre, de nombre DmxSimple. Esta se puede descargar desde la URL de Internet: **http://code.google.com/p/tinkerit/wiki/DmxSimple**

Para ello, la biblioteca se instala en el directorio de Arduino y se incluye en los programas utilizando la directiva:

```
#include <DmxSimple.h>
```

Para su programación, basta con usar sólo 3 funciones:

**a) DmxSimple.usePin(**<numero_Pin_Arduino>**)**

Indica qué pin de la tarjeta Arduino se utilizará para transmisión de datos hacia el DMX Shield. Si el valor es 0, se desactiva la biblioteca y el pin puede ser utilizado de forma normal con Arduino.

**b) DmxSimple.maxChannel(**<numero_máximo_canales_a_utilizar>**)**

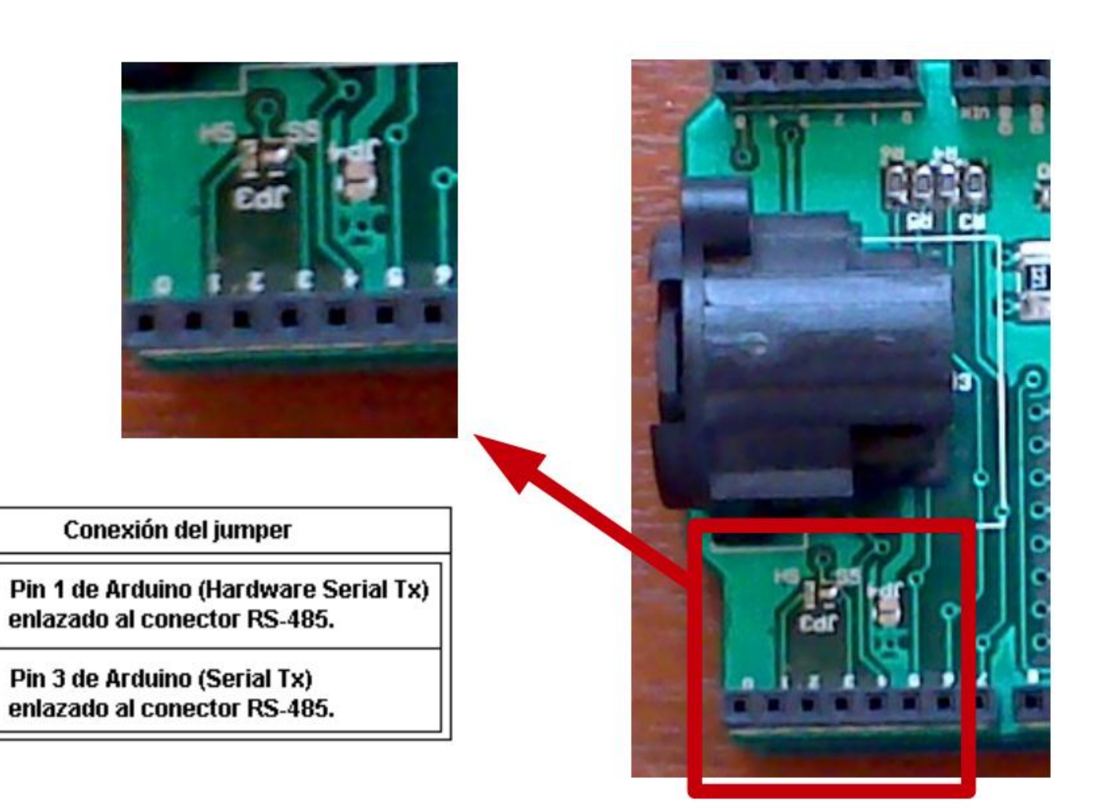
Indica cuál es el ID máximo de canal a utilizar mediante esta biblioteca. El número comprende cualquier valor entre 1 y 512. Mientras más bajo sea el número, más rápido es el envío de datos por la red, pero disminuye la cantidad de dispositivos que se podrán conectar al universo DMX.

**c) DmxSimple.write(**<número_canal>, <dato>**)**

Escribe un dato en ese canal. El número de canal puede ir entre 1 y 512, mientras que el dato puede variar entre 0 y 255.

En general, para la comunicación entre la placa Arduino y el DMX Shield se utiliza el pin 3. De todas maneras, cuando el shield se vende es necesario soldar un jumper que permite utilizar comunicación serie por hardware o software (véase la figura Nº 4.1). Las opciones son:

a) Comunicación hardware (HS - hardware serial): permite utilizar el módulo Serial convencional de Arduino para la comunicación RS-485 con el DMX Shield.
b) Comunicación software (SS - software serial): permite utilizar una biblioteca de software para Arduino para la comunicación RS-485 con el DMX Shield (como por ejemplo, DmxSimple o DMXSerial).



| Conexión del jumper | |
|---|---|
| HS | Pin 1 de Arduino (Hardware Serial Tx) enlazado al conector RS-485. |
| SS | Pin 3 de Arduino (Serial Tx) enlazado al conector RS-485. |

*Figura Nº 4: detalle del jumper HS/SS para selección de manejo del protocolo DMX-512 mediante el módulo Serial de hardware o una rutina de software para manejo serie a través de un pin de E/S común de Arduino. Nótese que para los ejemplos de este capítulo, el jumper está soldado hacia la derecha, activando el modo SS (de módulo serie en software, para utilizar la biblioteca DMXSimple).*

La soldadura debe hacerse desde el terminal central del jumper con el de la izquierda, para seleccionar el modo HS, o con el terminal de la derecha, para seleccionar el modo SS.

También se puede observar que en la placa se encuentran algunos controles extras que permiten realizar programas para Arduino y el DMX Shield que funcionen sin necesidad de usar un monitor serie o algún otro tipo de controlador gráfico. Estos controles de hardware son los siguientes (figura Nº 5):

a) Selector mecánico de 16 posiciones (DEVICE), para selección de 1 entre 16 dispositivos en una red DMX-512.
b) Selector mecánico de 16 posiciones (CHANNEL), para selección de 1 entre 16 canales de algún dispositivo DMX-512.
c) Potenciómetro (MASTER POT) para generar valores entre 0 y 255.

d) Botón (PRESET) para controlar algún tipo de evento en la placa.
e) Botón (RESET) para reiniciar la placa Arduino.

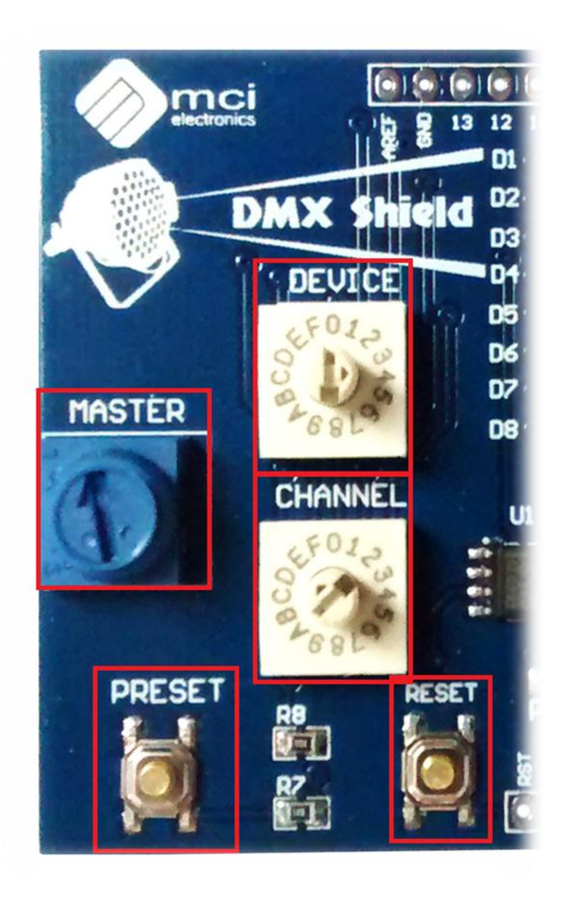



*Figura Nº 5: detalle de los controles de hardware que componen la placa DMX Shield.*

Para leer estos controles, se entregan a continuación los pines a los que está conectado cada control. Además, en el apéndice A, se entrega el detalle de una biblioteca de software que implementa funciones para leer estos datos, de manera automática.

a) El control MASTER está conectado al pin **A0** (pin analógico) de Arduino. Entrega, por ende, valores entre 0 y 1023.
b) El botón PRESET está conectado al pin **A5** (como pin digital, no analógico) de Arduino. Si se presiona, entrega un valor 0; si no, entrega un valor 1.
c) El botón RESET está conectado directamente al pin **RESET** de Arduino.
d) Tanto el selector DEVICE como el selector CHANNEL están conectados a los pines **D7**, **D5**, **D6** y **D4** (en orden de MSB a LSB). Si el selector DEVICE está configurado con el valor 10 (decimal), se entregarán los siguientes valores:

D7 = 1; D6 = 0; D4 = 1; D5 = 0

Es decir, el valor 1010 (binario).

e) Para leer el valor desde el selector DEVICE se debe escribir un valor HIGH en el pin **D8** de Arduino y luego leer los valores de los pines del punto (e); para hacerlo desde el selector CHANNEL, se debe escribir un valor **HIGH** en el pin **D9** de Arduino. Luego de leer el valor requerido, se debería escribir el valor **LOW** en dichos pines para evitar interferencia en la lectura entre ambos selectores.

Dados estos datos, se entiende que estos controles pueden ser utilizados en un programa de Arduino para realizar programas especiales que no necesiten el uso de un computador para la activación de los dispositivos DMX-512 y similares. Así, uno podría realizar programas como los siguientes:

a) Selección manual de un dispositivo DMX-512 en la red para aplicación local de efectos.
b) Selección manual de dispositivos y canales en dichos dispositivos para aplicación de efectos.
c) Creación de un programa para crear efectos automáticos almacenados en la EEPROM del Arduino.
d) Control de brillo manual o de tiempos entre efectos, utilizando el potenciómetro MASTER del DMX Shield.

Las posibilidades son numerosas y sólo dependen de la necesidad e imaginación del implementador del programa para Arduino y del dispositivo DMX-512 que se requiera controlar.

Junto con esta guía se entregan 9 ejemplos de utilización del DMX Shield, que indican cómo leer datos desde los controles de la placa y cómo aplicar los datos sobre el protocolo RS-485. Estos ejemplos se indican a continuación (tabla Nº 1):

| Código del Ejemplo | Descripción |
|---|---|
| **001** | Mostrar los colores rojo, verde y azul, con pausa de 1 [s] entre cada color. |
| **002** | Mostrar los 3 colores, cada uno con sus 256 niveles de brillo. |
| **003** | Mostrar los 3 colores con aumento y disminución de brillo. |
| **004** | Rotación de una serie de colores, indicados en un arreglo de colores. |
| **005** | Rotación de una serie de colores (como en el ejemplo 005), pero utilizando un efecto de brillo con aumento exponencial. |
| **006** | Efectos de mezcla de colores, con diferentes brillos, para obtener los colores de la rosa cromática. |
| **007** | Lectura de los valores de los controles del DMX Shield y muestra de dichos valores en la consola serie del Arduino. |
| **008** | Control de canal de colores y nivel de brillo, utilizando los selectores y el potenciómetro, para ver cómo trabajar con los valores en tiempo real. |
| **009** | Programa que permite la programación de una serie de efectos de colores y de pausas entre efectos, sólo utilizando los selectores, el potenciómetro y el botón *PRESET*. La secuencia quedará guardada dentro de la EEPROM del microcontrolador del Arduino (consulte el apéndice B para aprender a utilizar este programa). |

*Tabla Nº 1: listado de ejemplos para Arduino que acompañan esta guía.*

En general, para programar el protocolo DMX basta con seguir los siguientes pasos:

a) Informarse sobre los canales de cada dispositivo conectado a la red DMX-512, la función de cada canal y qué valores requieren para funcionar.
b) Unir los dispositivos en serie, según el plan de conexión diseñado.
c) Configurar el canal inicial (canal base o dirección base) para cada dispositivo DMX-512, conectado en la red.
d) Crear un programa de Arduino, inicializar el pin de comunicaciones con el DMX Shield e indicar la cantidad máxima de canales a utilizar.
e) Escribir diversos valores en diferentes canales, según la función que tenga cada canal en el dispositivo. Recordar que cada canal trabaja el mismo tiempo que los demás, por lo que es

posible generar efectos combinados entre varios canales (por ejemplo, activar al mismo tiempo los canales rojos y azules para generar tonos de morado o magenta, por ejemplo).

f) Probar el funcionamiento global del sistema.

## CONEXIÓN CON UN DISPOSITIVO DMX-512 Y EJEMPLO DE USO DE BIBLIOTECA DMXSIMPLE

Para el análisis siguiente, se empleará un foco de iluminación RGB disponible en el laboratorio, compatible con el protocolo DMX-512. El foco sólo dispone de 3 canales, los cuales permiten configurar el brillo de rojo (canal 1), verde (canal 2) y azul (canal 3) que desplegarán los LEDs frontales. El detalle del foco puede apreciarse en la figura Nº 6.




*Figura Nº 6: fotografía del foco de iluminación utilizado para los ejemplos de esta guía.*

El foco de iluminación posee en su parte trasera un pequeño visualizador y dos botones de ajuste. Este visualizador permite configurar el canal inicial (o dirección inicial) para este dispositivo DMX-512 dentro de la red. En la figura Nº 7, el foco se puede ver configurado para adoptar la dirección 015, dentro de las 512 posibles para la red.




*Figura Nº 7: fotografía del sistema de configuración de canal para el dispositivo de iluminación utilizado en los ejemplos de la guía.*

La imagen sólo es referencial; para los ejemplos de esta guía, el foco RGB utilizó al canal **001** como canal inicial. Por esta razón, el canal 1 del foco controlará el brillo del color rojo; el canal 2, el brillo del color verde; y el canal 3, el brillo del color azul.

Luego, se aplicó el programa para Arduino que se mostrará a continuación en la figura Nº 8. Este programa (el ejemplo 001 que acompaña a esta guía) sólo rota los tres colores primarios en orden, con una pausa de 1 segundo entre cada cambio de color.

```
#include <DmxSimple.h>

#define CANAL_ROJO 1
#define CANAL_VERDE 2
#define CANAL_AZUL 3
#define MAX_NUM_CANALES 3

void setup()
{
 DmxSimple.usePin(3);
 DmxSimple.maxChannel(MAX_NUM_CANALES);
}
```

*Figura Nº 8: inicialización para el programa de rotación de colores (ejemplo 001).*

Nótese que primero se debe incluir la biblioteca DmxSimple.h para acceder a usar las funciones del protocolo DMX-512. Luego, de forma opcional, se definen algunos nombres simbólicos para denominar a cada uno de los canales del dispositivo, se indica el pin de Arduino por el que se realizará la comunicación serie y se indica la cantidad máxima de canales a soportar en la red.

```
void loop()
{
 DmxSimple.write(CANAL_ROJO ,255);
 DmxSimple.write(CANAL_VERDE,0);
 DmxSimple.write(CANAL_AZUL ,0);
 delay(1000);

 DmxSimple.write(CANAL_ROJO ,0);
 DmxSimple.write(CANAL_VERDE,255);
 DmxSimple.write(CANAL_AZUL ,0);
 delay(1000);

 DmxSimple.write(CANAL_ROJO ,0);
 DmxSimple.write(CANAL_VERDE,0);
 DmxSimple.write(CANAL_AZUL ,255);
 delay(1000);
}
```

*Figura Nº 9: ciclo infinito para rotar los colores de forma indefinida. Cada color se visualiza durante 1 segundo en el foco de iluminación.*

Luego, se observa que en la segunda parte del código (figura Nº 9) se va realizando una llamada a la función DmxSimple.write( ) para escribir valores en cada canal del foco RGB. Se debe recordar que para generar colores en un foco RGB es necesario combinar 3 tonalidades (o brillos) de colores primarios: rojo, verde y azul. Cero indica ausencia de brillo, mientras que el valor 255 indica el máximo de brillo posible para el dispositivo. De esta forma, en cada bloque de 3 funciones (entre cada función delay( ) ) se observa que se despliegan los siguientes colores:

a) **255** para rojo, 0 para verde y 0 para azul => se despliega el color **rojo fuerte**.
b) 0 para rojo, **255** para verde y 0 para azul => se despliega el color **verde fuerte**.
c) 0 para rojo, 0 para verde y **255** para azul => se despliega el color **azul fuerte**.

Para ver el resultado final, basta con montar el DMX Shield sobre la placa Arduino, conectar la placa Arduino mediante su cable USB al computador (para encenderla), grabar el programa compilado en su memoria, encender el foco RGB y conectar su cable de red a la entrada en el DMX Shield. El resultado de ejecutar tal programa de forma indefinida puede apreciarse en la figura Nº 10.

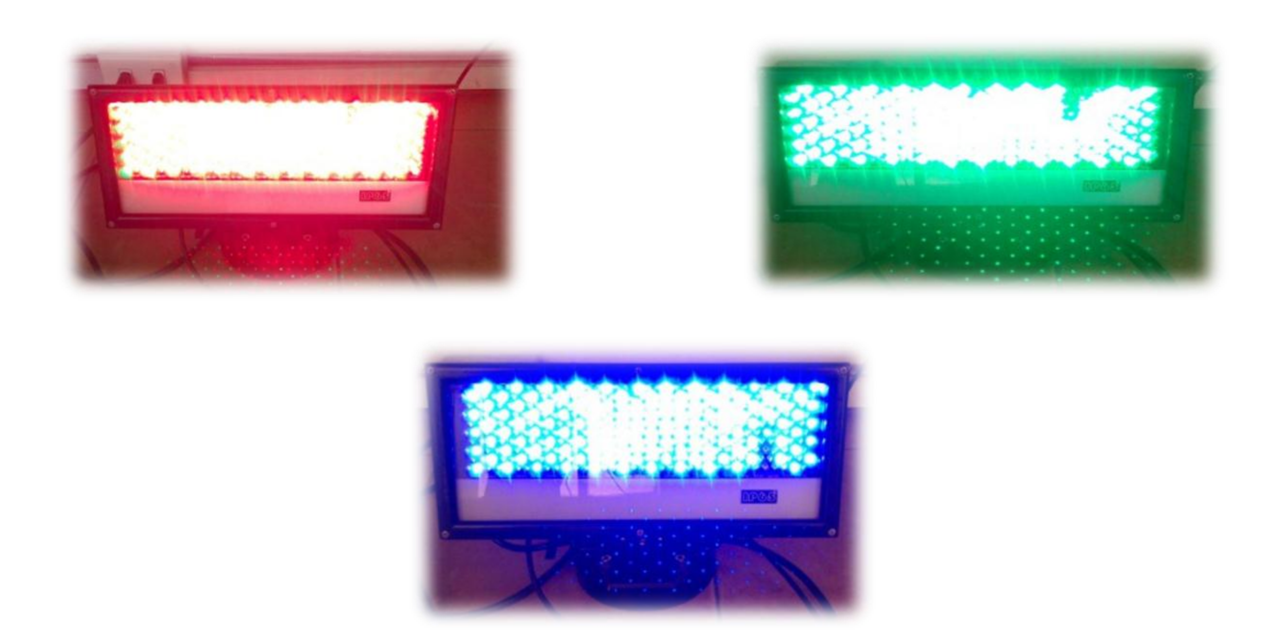
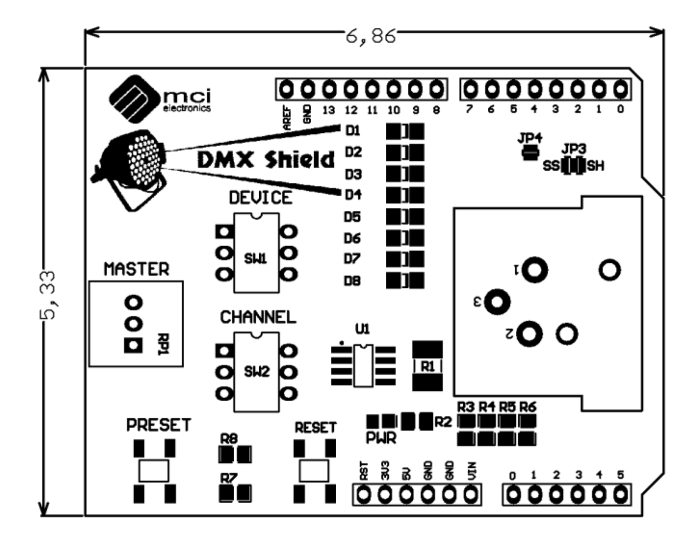


*Figura Nº 10: ciclo de colores como resultado a la ejecución del programa antes mostrado.*

## CARACTERÍSTICAS MECÁNICAS

Las dimensiones son: 5.33cm x 6.86cm



## APÉNDICE A: CÓDIGO FUENTE DE LA BIBLIOTECA DMXSHIELD.CPP

A continuación se entrega el código de la biblioteca DmxShield.cpp, que incluye funciones para leer los controles de la placa DMX Shield de MCI Electronics.

```
#include <Arduino.h>
#include <DmxSimple.h>
#include "DmxShield.h"

void DmxShieldInit()
{
	pinMode(PIN_PRESET, INPUT);
	pinMode(PIN_SELECTOR_0, INPUT);
	pinMode(PIN_SELECTOR_1, INPUT);
	pinMode(PIN_SELECTOR_2, INPUT);
	pinMode(PIN_SELECTOR_3, INPUT);
	pinMode(PIN_SELDEVICE, OUTPUT);
	pinMode(PIN_SELCHANNEL, OUTPUT);
	pinMode(PIN_MASTER, INPUT);
}
byte readPresetButton() { return digitalRead(PIN_PRESET); }
int readMasterPot() { return analogRead(PIN_MASTER); }
byte readDeviceSelector()
{
	byte d;
	digitalWrite(PIN_SELDEVICE, HIGH);
	d = digitalRead(PIN_SELECTOR_3) << 3 | digitalRead(PIN_SELECTOR_2) << 2 |
	digitalRead(PIN_SELECTOR_1) << 1 | digitalRead(PIN_SELECTOR_0);
	digitalWrite(PIN_SELDEVICE, LOW);
	return d;
}
byte readChannelSelector()
{
	byte d;
	digitalWrite(PIN_SELCHANNEL, HIGH);
	d = digitalRead(PIN_SELECTOR_3) << 3 | digitalRead(PIN_SELECTOR_2) << 2 |
	digitalRead(PIN_SELECTOR_1) << 1 | digitalRead(PIN_SELECTOR_0);
	digitalWrite(PIN_SELCHANNEL, LOW);
	return d;
}
```

Las definiciones de pines se entregan en el archivo de cabecera DmxShield.h.

```
#include <Arduino.h>

#define PIN_PRESET 19 // Pin analógico A5
#define PIN_SELECTOR_0 4
#define PIN_SELECTOR_1 6
#define PIN_SELECTOR_2 5
#define PIN_SELECTOR_3 7
#define PIN_SELDEVICE 8
#define PIN_SELCHANNEL 9
#define PIN_MASTER 0 // Pin analógico 0
```

La lista de funciones implementada en la biblioteca se muestra en la tabla Nº 2:

| Función implementada | Descripción y valores devueltos |
|---|---|
| `void DmxShieldInit()` | Inicializa los pines de los controles de la placa DMX Shield. |
| `byte readPresetButton()` | Devuelve 0 si se presionó el botón *PRESET* ó 1 si el botón está liberado. |
| `int readMasterPot()` | Devuelve un valor entero entre 0 y 1023 para indicar la lectura del potenciómetro *MASTER*. |
| `byte readDeviceSelector()` | Devuelve un valor entre 0 y 15 para el selector *DEVICE*. |
| `byte readChannelSelector()` | Devuelve un valor entre 0 y 15 para el selector *CHANNEL*. |

*Tabla Nº 2: inicialización para el programa de rotación de colores (ejemplo 001).*

## APÉNDICE B: FORMA DE OPERACIÓN (EJEMPLO 009)

A continuación se entrega un manual resumido para aprender a utilizar el programa dado en el ejemplo 009. Este es un programador básico de secuencias de luces que permite que la placa Arduino tome el control de una red DMX-512 y de los focos de luces RGB de esta red. Sus limitaciones son las siguientes:

a) Sólo permite controlar hasta 15 dispositivos, de hasta 15 canales cada uno.
b) Los dispositivos deben tener direcciones consecutivas, por la limitante señalada en el punto anterior.
c) Permite grabar la secuencia de luces en la EEPROM del microcontrolador de la placa Arduino, pero sólo 1 secuencia. Para otra secuencia, debe borrarse la anterior.

El programa deberá modificarse o reescribirse, de acuerdo a las necesidades de cada usuario. Recuérdese que este programa sólo es una muestra de las múltiples posibilidades de implementación basadas en el DMX Shield.

- **Encendido y modos de operación**

El programa convierte al Arduino en un controlador DMX-512. Este controlador tiene 2 modos de operación: programación de secuencia de luces y reproducción de una secuencia ya escrita.

- Para ingresar al modo de programación:
  - Conecte el DMX Shield al Arduino y éste a su fuente de alimentación (cable USB, por ejemplo).
  - Mantenga presionado el botón PRESET y luego presione el botón RESET de la placa.
  - Luego de 3 segundos presionando, suelte el botón PRESET.
  - Finalmente, las luces de todos los focos RGB se apagarán. Ya está en el modo de programación.
- Para ingresar al modo de reproducción, luego de tener una secuencia grabada:
  - Sólo encienda la placa Arduino con el DMX Shield montado sobre el Arduino. Si hay alguna secuencia en la EEPROM, se ejecutará automáticamente.

- **Encendido y modos de operación**

El programa dado asume que cada foco RGB sólo tiene 3 canales como máximo (canales de rojo, verde y azul) y que hay un máximo de 3 canales en la red (es decir, un único foco RGB). Deben cambiarse los valores de las directivas #define para redefinir este comportamiento, en las líneas 26 y 27 del código:

```
...
#define TOTAL_CANALES_RED 3
#define MAX_NUM_CANALES_POR_DISPOSITIVO 3
...
```

Al estar en el modo de programación se pueden seguir los siguientes pasos:

a) Para programar un canal, seleccione el número del dispositivo con el selector DEVICE (entre 1 y 15) y el número del canal a modificar (entre 1 y 15). Luego, mueva el potenciómetro MASTER para seleccionar un valor de brillo entre 0 y 255, que se podrá ir visualizando en tiempo real sobre el foco RGB seleccionado.
b) Luego, al tener el color requerido, mantener presionado el botón PRESET por 3 segundos y luego soltarlo. El color ha quedado grabado en la EEPROM.
c) Después, continuar seleccionando otros dispositivos, canales y valores para agregar más colores a la secuencia.
d) Cuando se agrega un color a continuación de otro, estos se mostrarán en forma simultánea. Para agregar una pausa entre un color y otro se pueden usar los eventos de pausa. La pausa en centésimas de segundo se activa con el selector DEVICE en 0 y el selector CHANNEL en la posición 1. Luego, seleccione con el potenciómetro el valor de pausa (entre 0 y 2,55 segundos) y grábelo en memoria, según el paso (b).
e) También existe una pausa de décimas de segundo. Esta se activa con el selector DEVICE en 0 y el selector CHANNEL en la posición 2. Luego, seleccione con el potenciómetro el valor de pausa (entre 0 y 25,5 segundos) y grábelo en memoria, según el paso (b).
f) Finalmente, cuando se haya ingresado el último evento de la secuencia, lleve el selector DEVICE a 0 y el selector CHANNEL a la posición 0 y siga el paso (b). Con esto se da finalizada la secuencia de colores y esta se reproducirá automáticamente al reiniciar la placa Arduino sin presionar ningún botón.
g) Por último, si se quiere borrar todo el programa en la EEPROM, lleve el selector DEVICE a 0 y el selector CHANNEL a la posición F (posición 15) y siga el paso (b). Esto eliminará la secuencia hecha y permitirá guardar otra nueva.

## HISTORIA DEL DOCUMENTO

| Revisión | Fecha | Editado por | Descripción/Cambios |
|---|---|---|---|
| 1.0 | 27 de Enero de 2014 | Manuel López | Versión inicial del documento |
| 1.1 | 06 de Agosto de 2014 | Diego Muñoz | Revisión y corrección |